

Vertraulich

An
Siemens AG

ERFINDUNGSMELDUNG
Bitte verschlossen weitersenden!

Mail BN 75

Aktenzeichen der GR PA

GE 8583

Ich/Wir (Voramen u. Name der/des Erfinder/s, weitere Angaben u. Unterschrift/en auf Seite 4)

Dr. Douglas C. Schmidt, U. Dorn, D. Quell, D. Becker,
C. Scharf

Datum der Ausfertigung

31.5.96

melde/n hiemit die auf den Seiten 2 und 3 vollständig beschriebene Erfindung mit der Bezeichnung:

INDIGO - Interpretative Network Daemon Implemented by means of
a GENERIC-Main Object

I. An Vorgesetzten des/der Erfinder/s

Dienststelle

Eingang am:

Herrn/Frau Dr. Wangler

BSW

mit der Bitte, die nachstehenden Fragen zu beantworten:

- a) Wann ging die Erfindungsmeldung bei Ihnen ein? → → → → → → → → → →
b) Geht die Erfindung auf öffentlich geförderte Arbeiten zurück?

☒ nein ☐ ja, Projekt (Vorhaben)

Ab Eingang läuft gesetzliche Frist!

1.7.96

(Datum) (Unterschrift des Vorgesetzten)

Bitte bei Zuständigkeit auch
zu Ziffer III. Stellung nehmen.

II. Bitte wegen gesetzlicher Frist sofort weiterleiten:

An Patentabteilung (ZFE GR PA)

zur weiteren Veranlassung.

Eingang am:
ZFE GR ZD VE

Eing. 24. JULI 1996

GR

III. An Werks- bzw. Abteilungsleitung

Eingang am:

BSW

Herrn/Frau

Dr. Wangler

zur Entscheidung bzw. Empfehlung über Inanspruchnahme (Zutreffendes bitte ankreuzen!):



Die Erfindung sollte unbeschränkt in Anspruch genommen werden
Kosten trägt (Werk/Abt., Konto): BSW 362



Die Erfindung kommt für eine Behandlung als Betriebsgeheimnis in Betracht.



Die Erfindung kommt evtl. für Auslandsanmeldungen in Betracht.



Die Erfindung wird voraussichtlich nicht benutzt.
Bei Freigabe wäre aber ein Benutzungsrecht wünschenswert.



Die Erfindung kann dem/den Erfinder/n vorbehaltlos freigegeben werden.



Die Erfindung betrifft nicht unser Interessengebiet. Es sind noch folgende
Dienststellen zu befragen:

Vermerke der ZFE GR PA

* Bitte vertragliche
vereinbarung mit
Dr. Schmidt treffen
u. ggf. Bestätigung
der Erfindungsmeldg.

BSW
30.7.96

19.07.96

Wangler

(Datum) (Unterschrift der Werks- bzw. Abteilungsleitung)

Dringlichkeitsvermerk

IV. Zurückerbeten an ZFE GR PA

APPENDIX A

1. Welches technische Problem soll durch Ihre Erfindung gelöst werden? ^{Main() - Programm of a running Executable}
2. Wie wurde dieses Problem bisher gelöst? ^{only source code approach known!}
3. In welcher Weise löst Ihre Erfindung das angegebene technische Problem?
^{See appendix!}

The principal new approach is delivering only a binary executable including all of the following features:

a) Objectoriented

b) proper hidden instantiation of processwide Singleton-objects e.g. for:

- basic network-communication features with ATOMIC
(see ATOMIC Erfindungsgeldung 31.5.95
GR 95E3632DE)

- basic sync/async Management with SESAM
(see SESAM Erfindungsgeldung 31.5.95
GR 95E3631DE)

- basic dynamic linking features for Component
Dynamic Link Libraries (DLLs)
(see Service Configuration Pattern,
Douglas C. Schmidt)

- basic Networkwide Naming Support

- basic Networkwide Time Share Support

- basic Networkwide Resource locking Support

- basic Networkwide Message logging Support

- basic Operating System Abstraction Layer (OSAL)

- basic interface to a System Configuration Control (see)

c) Support of full duplex Event and Request/
Response Channels

d) Generic Connection to dominant
full-Framework supported Main() - Programms
(Message Pump Interconnection Protocol (MIP)).

e) Generic Connection to Device - Drivers

f) Generic Support of an Object - Dir -
Database (Debugging Port)

(see External Polymorphism pattern, Douglas C. Schmidt)

This is a completely Domain - Software independent main
module, which can be dynamically configured by
Domain - Configuration

INDIGO - Interpretative Network Daemon Implemented by means of a GENERIC- Main Object

The principal new approach is delivering only a binary executeable including all of the following features:

- a) object oriented
- b) proper hidden instantiation of process-wide Singleton-objects e.g. for:
 - basic network-communication features with ATOMIC (see ATOMIC Erfindungsmeldung 31.5.95, GR 95 E 3632 DE)
 - basic sync/async Management with SESAM (see SESAM Erfindungsmeldung 31.5.95, GR 95 E 3631 DE)
 - basic dynamic linking features for component Dynamic Link Libraries (DLLs)
(see Service Configuration Pattern, Douglas C. Schmidt)
 - basic Networkwide Naming Support
 - basic Networkwide Time Base Support
 - basic Networkwide Resource Looking Support
 - basic Networkwide Message Logging Support
 - basic Operating System Abstraction Layer (OSAL)
 - basic interface to a system configuration Control (SCCM)
- c) Support of fullduplex Event and Request /Response Channels
- d) Generic Connection to dominant GUI-Framework Supported Main ()-Programms (Message Pump Interconnection Protocol (MPIP))
- e) Generic Connection to Device-Drivers
- f) Generic Support of an Object-...-Database (Debugging Port) (see External Polymerphil pattern, Douglas C. Schmidt)

This is a completely Domain-Software independent main Module, which can be dynamically (re-) configured by Domain-component-DLLs at Runtime.

INDIGO -

A Generic Main Approach

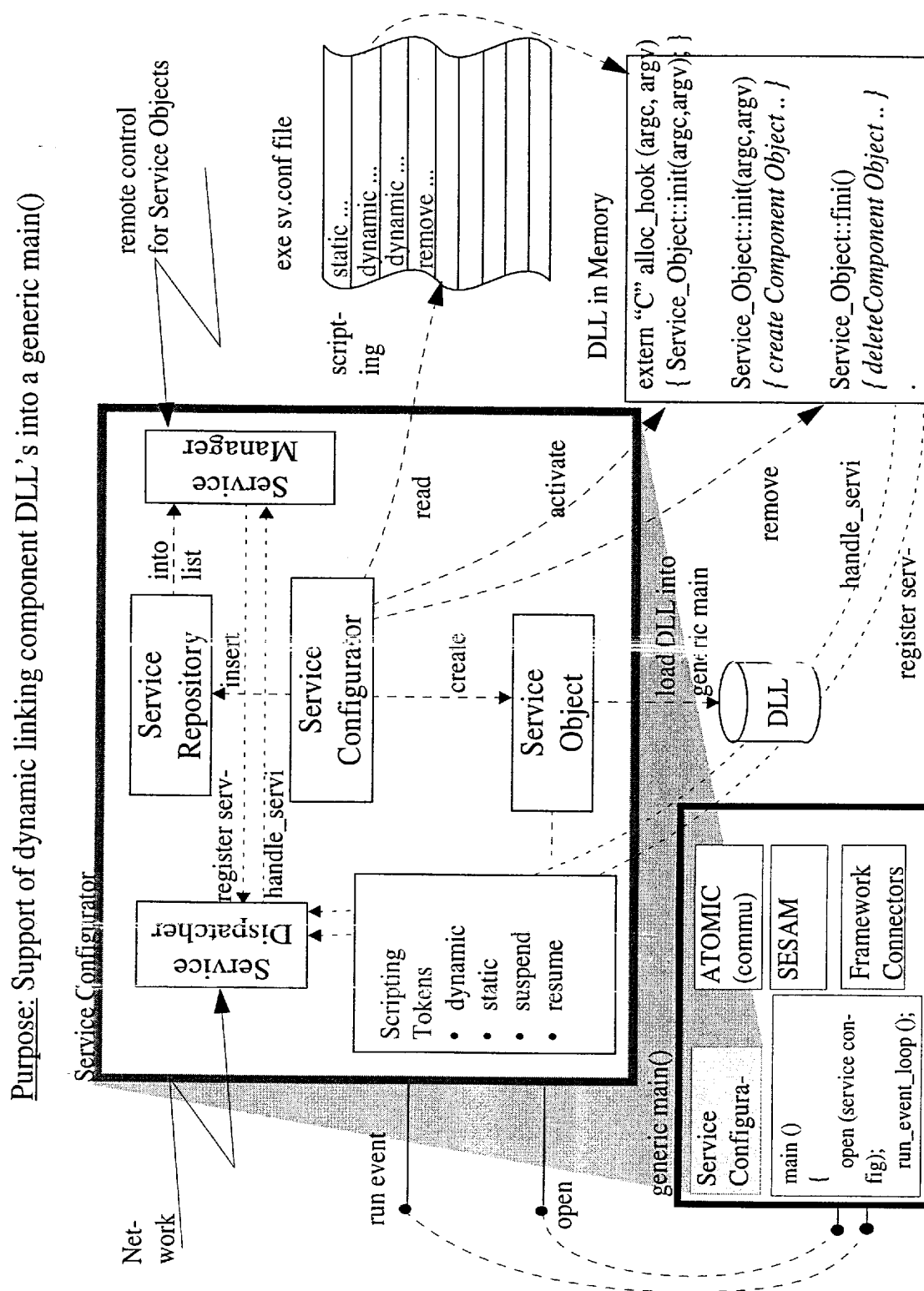
Indigo is based on the Service Configuration pattern.

The Service Configurator pattern offers the following benefits:

- **Centralized administration:** The pattern combines one or more services to a single administrative unit. This simplifies development by automatically performing common service initialization and termination activities (eg., opening and closing files). It also makes it possible to centralize the administration of network services by using a uniform set of configuration operations such as *initialize*, *suspend*, *resume* and *terminate* and query application component.
- **Increased modularity and reuse:** The pattern helps to decouple the implementation of services from the configuration of these services and thus improves the modularity and reusability of those. As said above, all services have a uniform interface by which they are configured. This also encourages reuse.
- **Increased Configuration dynamism:** The pattern makes it possible to dynamically reconfigure a components without modifying, recompiling or relinking existing code. In addition to that it is possible to reconfigure a component or other components without influencing other co-located components.
- **Increased opportunity for tuning and optimization:** The pattern moves up the range of configuration alternatives available to developers. Services functionality is decoupled from the execution agent (the Indigo generic main) used in order to invoke the service. Therefore developers can adaptively tune daemon concurrency levels to meet client demands. Some alternatives include spawning a thread or process at the arrival of a client request or at creation time.
- **Increase security of executables in avoiding the usage of process global data.** Users can bring in functionality into the Generic Main only in components, which have a set of initialisation and finalisation hooks.

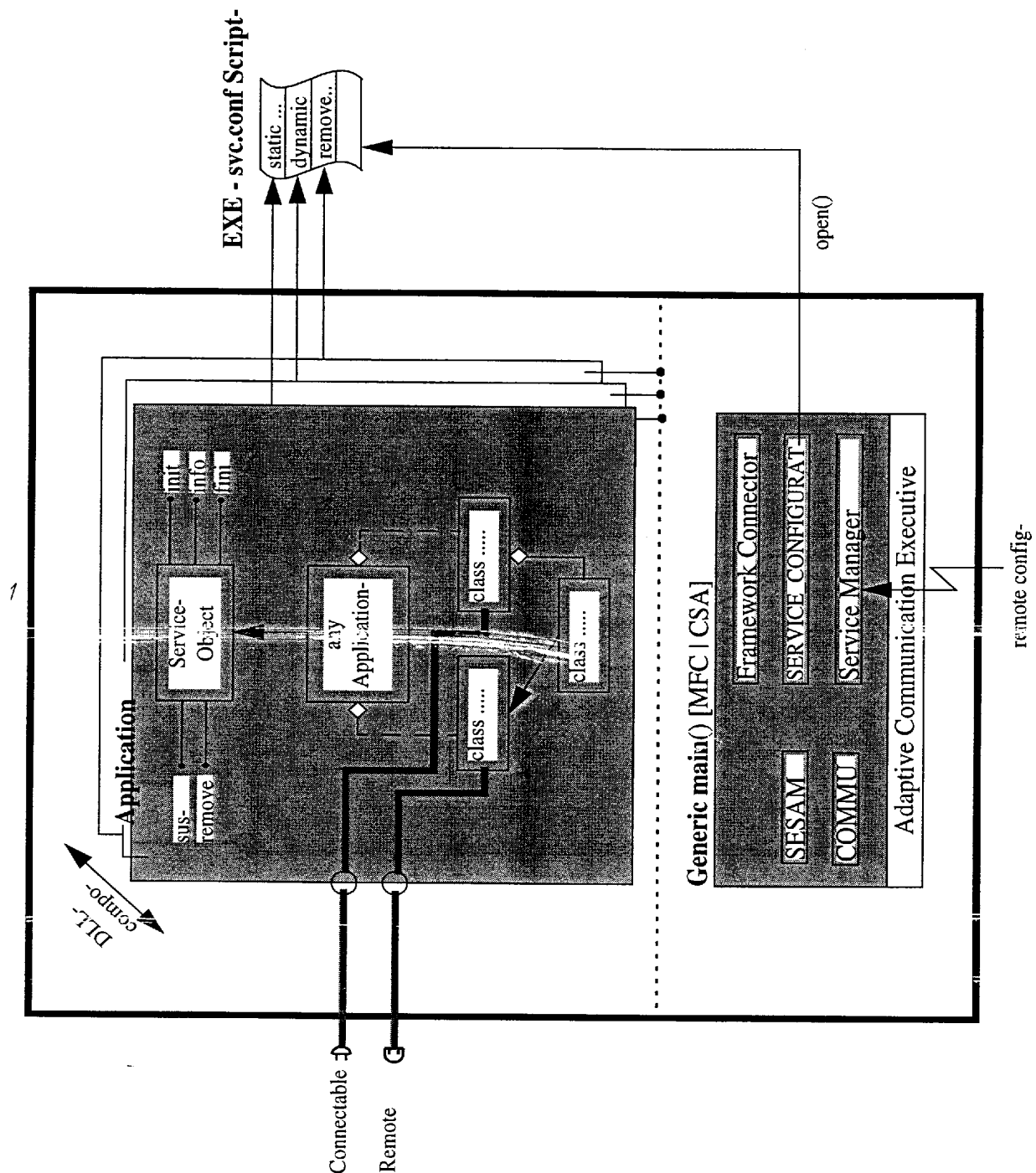
The following pictures show the principals.

1 Service Configurator Pattern



Copyright © Siemens AG 1996. All rights reserved. For internal use only.
Alle Rechte vorbehalten. Nur für internen Gebrauch.

2 Executable Template



The following figure shows how the Generic Main is enriched by a Sync/Async Management pattern (SESAM)

5 Generic Main and MFC-ACE Framework Interaction - reflective Adaption of the Generic Main as a connector to 3rd part Main-Moduls

There are two goals of the Generic Main:

- Interconnecting Frameworks (ACE and MFC)
- Creating a main-program which can never be changed by an application programmer

5.1 Interconnecting Frameworks

The following figure shows the coupling of the MFC and the ACE Main Message Pumps. Every Thread has got its own message pump. The message pump of the MFC is hidden behind the scenes, the message pump of the ACE is attached to an ACE_Task object.

5.1.1 MFC->ACE Communication

An abstract base class implements a generic main base functionality which could be extended by deriving from this class and adding functionality.

Especially the both hook-functions of the Generic Main base class are very helpful. You could place code before the dispatch loop will start and the second hook will be called after the dispatch loop is ready.

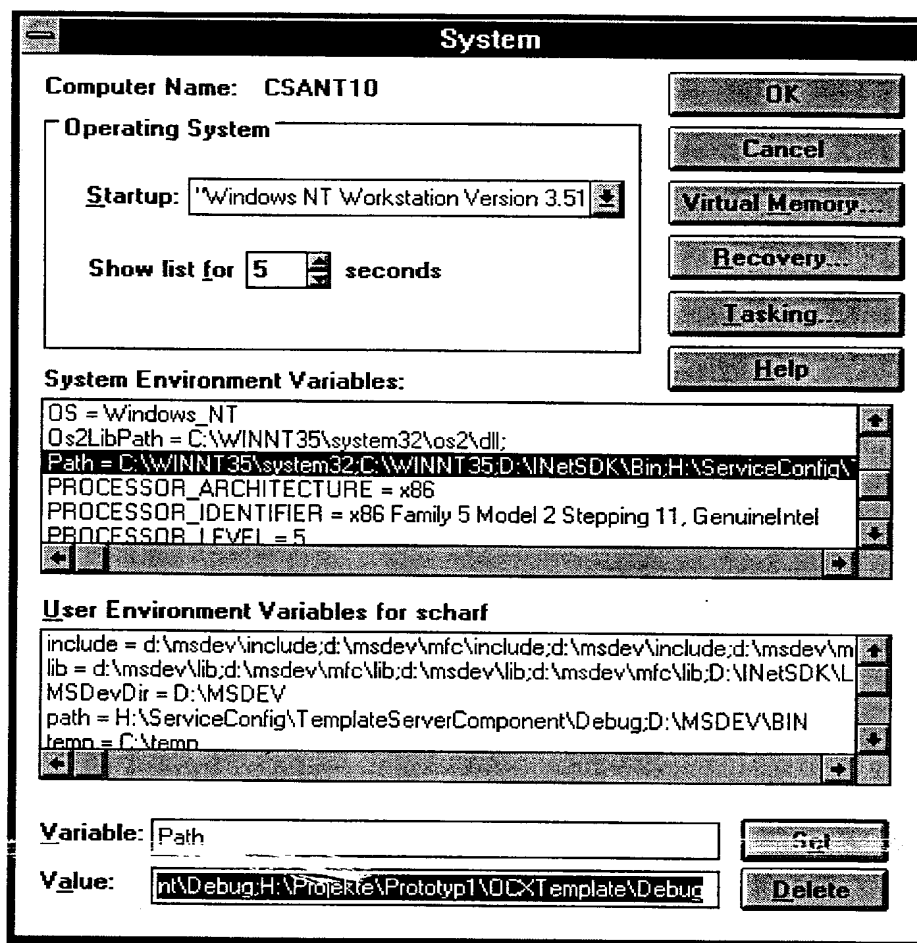
This way we use functionality from the MFC framework. The entry point here is the Post-Message method.

The notification of this method looks like:

```
BOOL PostMessage(  
  HWND hWnd, // handle of destination window  
  UINT Msg, // message to post  
  WPARAM wParam, // first message parameter  
  LPARAM lParam // second message parameter  
);
```

What is the meaning of the parameters?

- **HWND hWnd:** This is the handle of the recipient window
- **UINT msg:** This is the identifier of the user defined message..
- **WPARAM wParam:** First parameter of the message.
- **LPARAM lParam:** Second parameter of the message. Used to transfer the name of the loaded components from ACE to MFC.



The runtime linker uses the path environment variable to search for DLLs and OCXs. Thus it is required to enter the path of the component into the path-variable.
This window is part of Main/Control Panel/System of the Microsoft-NT platform.

The following figure shows a running Machine snapshot, where different Generic Mains are running, configured by various component DLLs.

7 Application Templates to support Service Configuration into Generic Main

Modern operating systems, e.g. Windows NT, provide support for dynamically configurable kernel-level device drivers. Similar to that, CSA provides different program components in OCX format. These can be linked into and unlinked out of the application dynamically. This makes it possible to reconfigure the application without having to recompile and relink new components into the application.

The way this is achieved is by the use of the *Service Configurator Pattern*. The Service Configuration pattern resolves the following issues:

- *The need to defer the selection of a particular type, or a particular implementation, of a service until very late in the design cycle* - This allows developers to concentrate on the functionality of a service, without committing themselves prematurely to a particular service configuration. By decoupling functionality the Service Configuration pattern permits applications to evolve independently of the configuration policies and mechanism by the system.
- *The need to build complete applications or systems by composing multiple independently developed services* - The Service Configuration pattern requires all services to have a uniform interface. This allows the services to be treated as building blocks which can be easily put together as components of a large application. The uniform interface across all the services makes them "look and feel" the same with respect to how they are configured and this makes developing applications simpler.
- *The need to optimize, reconfigure and control the behavior of a service at run-time* - Decoupling the implementation of a service from its configuration makes it possible to fine-tune certain implementation or configuration parameters of services. For instance, depending on the parallelism available on the hardware and operating system, it may be more or less efficient to run one or more services in separate threads or processes. The Service Configuration pattern enables applications to control these behaviors at run-time, when more information may be available to help optimize the services.

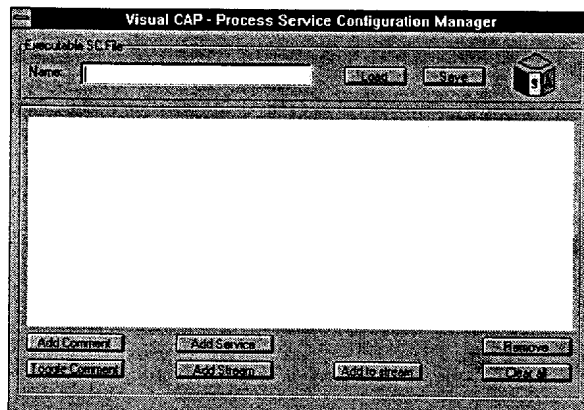
(from P. Jain and D.C. Schmidt, Service Configurator, A Pattern for Dynamic Configuration and Reconfiguration of Communication Services)

The *Service Configuration framework* uses a configuration file (e.g. *svc.conf*) to guide the configuration and reconfiguration activities of a process. Each of these processes has therefore to be associated with a distinct configuration file.

Each *service config entry* in the file begins with a *service config directive* that specifies the action to perform. Each entry contains certain attributes that indicate the location of the shared object file for each dynamically linked object and parameters as well. The latter may be needed to initialize the service at runtime.

8 Program description of SVCman

The Service Configuration Manager is a software-tool that allows you to create and modify your Service Config files in a comfortable way.



According to the Service Configurator Framework the SVCman supports the following Service Config Directives:

symbol	description
dynamic	Dynamically link and enable a service
static	Enable a statically linked service
remove	Completely remove a service
suspend	Suspend service without removing it
resume	Resume a previously suspended service
stream	Configure a stream into a daemon
string	Configure a remote facility

Table 1: Service Config Directives

In addition to that the SVCman is also designed to work with a ProcessManager, that allows you to define and change the configuration for a specific process dynamically.

If the corresponding process of the current Service Config file is not running, this is not a problem at all. In this case the Service Config file will only be written to disk and the changes take effect when the process is started the next time. Nevertheless the old file will be saved as *.old file.

If the process is currently running and the changes should take effect in advance one has to develop an alternative. The SVCman then generates a Delta file (*.delta) which holds the necessary directives to bring the Service Configuration Framework to the changed status. When the process is started the next time, it will be of course defined corresponding to the new Service Config file.

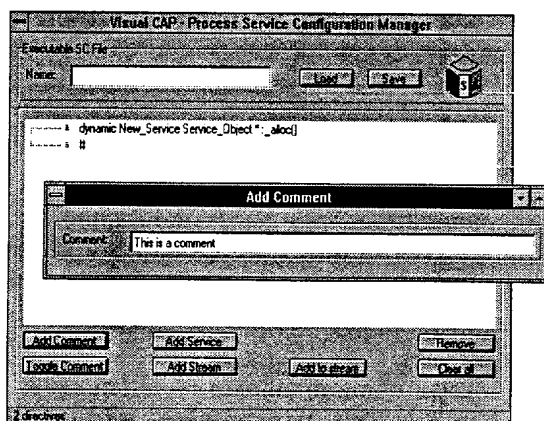
Entry	Description
Function	Name of function of the service entry in the DLL
Status	Status of service (active, inactive)
Parameter	optional parameterstring

Table 2: Description of entry-fields

- Note that the Enter key will act as the Window-Close button and thus close the form. You do not have to press Enter at the end of each entry-field. The changes you make will be updated in any case.
- The Browse button helps you to find the correct name of the library by showing you a list of the currently available ones in the specified path.
- According to naming conventions there must not be blanks (SPACE) inside name-fields, these are therefore depressed.
- As the Service Configurator Framework uses double quotes (") to indicate the start and the end of a parameter there must not be double quotes inside the parameter-field. They are therefore depressed.
- For a correct definition of a service all fields in the form have to be filled. Several types of services do require only one or two entries. The not required ones are disabled by program.
- During editing the Service Property form, the main form is disabled.

9.2.2 Add Comment

With Add Comment you can simply mark a line for comment and thus give some remarks on the entries of the config-file. This clearly enhances the readability of the file.



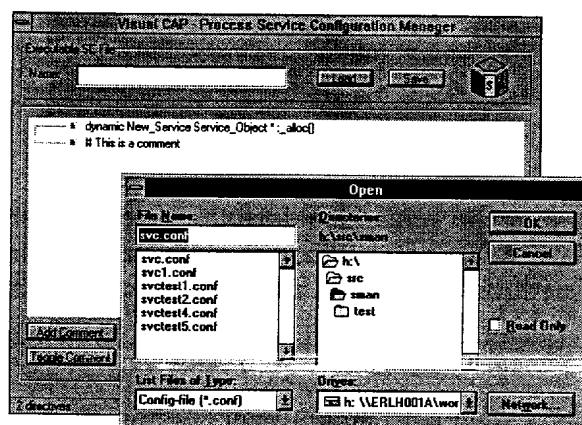
9.2.8 Clear all

Use this Button to clear all entries and reset all.

- Note that there is no UNDELETE, so be careful.
- The filename will also be cleared and you will have to define a new one when you want to save the file.

9.3 ... Save a config file

After editing or changing the file you can save your config file to disk. A common Windows Dialogbox will be opened and you have to specify the path and filename.



- The file will be stored as `svc.conf` by default.
- If you have specified a different filename, the executable has to have the `-f FILEPATH-FILENAME` given or a Process Start Parameter
- If there is an existing file on disk it will be renamed into `*.old`, so you will not loose it and can restore it

Note:

As said above, the SVCman is also designed to work with a ProcessManager, that allows to define the configuration for a specific process dynamically. If therefore the corresponding process of the changed Service Config file is running, you will be asked if the changes you made should take effect in advance or not.

```

/*{ Compilation unit -----
Component      : CSA -
Name           : CsaGenericMainBas.h
Author        : (D. Quehl, BNEP1, +49 9131 84 2430); Siemens AG B Med CSA
Language      : C/C++
Creation Date  : 19-JUN-1996
Test State    :
Description    : Definition of Generic Main functionality.
Requirement Key : os_process_mqjr

```

Copyright (C) Siemens AG 1995 All Rights Reserved

```

/*} END */
#include <ace/OS.h>
#include <ace/Task.h>
#include <ace/Synch.h>
#include <ace/Synch_T.h>
#include <CsaCommon/CsaDefs.h>
#include <osc/CsaOscNlsMessageId.h>

```

```

#pragma comment ( exestr, "e(1) CsaGenericMainBas.h 1.2 (10 Jul 1996) : Definition Generic Main Base class" )

```

```

/*{ Data type -----
Name           : CSA_STATUS - macro for Msc component
Description    : Macro for error Handling
                -- uses status as Input
                -- hides setting of Line and File

```

```

#include CSA_ERR_DISABLE
#include <err/CsaStatus.h>
#define CSA_STATUS_gmain(status) | csaStatus->set(status, _FILE_, __LINE_);
false
#define CSA_STATUS_gmain(status);
endif // #ifndef CSA_ERR_DISABLE
#define CSA_STATUS_GMAIN(status) CSA_STATUS_gmain(CSA_STATUS_MSC_##status)

```

```

/*} END Data type */

/*{ Data type -----
Name           : CSA_TRACE_IN - Macro redefinition.
Description    : Macro redefinition that disables method trace.

```

```

#include CSA_TRACE_DISABLE
#include <err/CsaTrace.h>
false
#define CSA_TRACE_IN(a)
endif // #ifndef CSA_TRACE_DISABLE
/*} END Data type */

```

```

/*{ Class -----
Name           : CsaGenericMainBas
Description    : CsaGenericMainBas provides the generic main program
                functionality. CsaGenericMainBas is derived from
                ACE_Task<ACE_MT_SYNCH> to let the dispatch loop
                run in either, the main (caller) thread or a
                separate thread (task's svc method).

```

```

class CsaGenericMainBas : public ACE_Task<ACE_MT_SYNCH> {
public:
    // Constructor/destructor
    CsaGenericMainBas(void);
    CsaGenericMainBas(int argc, char *argv[]);
    ~CsaGenericMainBas(void) {
        delete this;
    };
    // Store commandline arguments if instantiated using the default
    // constructor.
    void setArgs (int argc, char *argv[]) {
        argc_ = argc;
        argv_ = argv;
    };
    // handle SIGINT
    virtual int handle_signal (int signum);
    // Perform initialization steps for CsaGenericMainBas

```



```

virtual int open (void *p = 0);

// Invoke dispatch loop [a]synchronously.
virtual int dispatchLoop (bool async = false);

// synchronize on dispatcher termination
virtual void sync(void);

// break the loop
virtual int endloop(void);

// hook methods that can be overloaded to insert functionality before
// the loop starts and immediately after the loop finished.
virtual int looplook1(void) { return 0; }
virtual int looplook2(void) { return 0; }

// hook methods that can be overloaded to insert functionality prior
// to demon open and after demon close.
virtual int openlook(void) { return 0; }
virtual int closelook(void) { return 0; }

private:

// The implementation of the main dispatch loop.
virtual int svc (void);

// The following methods are inherited from ACE_Task which defines
// them as pure virtual methods.
// They are not needed here and implemented returning instant error.
// Subclasses of CsaGenericMainBas may overload them.
virtual int put (ACE_Message_Block *, ACE_Time_Value * = 0) {
    return -1;
}
virtual int close (unsigned long flags = 0) {
    return 0;
}
virtual int handle_input (ACE_HANDLE fd = ACE_INVALID_HANDLE) {
    return -1;
}

// The service configuration daemon
ACE_Service_Config daemon_;

// Copy of the command line arguments
int argc_;
char **argv_;

// Indicator for initialization state
bool initialized_;

// API lock that prevents loop from being called multiply
// and synchronizes the termination of the thread that runs
// the loop asynchronously with the caller thread.
ACE_Thread_Mutex lock_;

// provide ACE_task a thread manager

```

```

ACE_Thread_Manager *thMgr_;

};

/*} END Class */

#endif //ifndef CSAGENERICMAINBAS_H

/*{ Change header -----
Date       : 26. Jun 1996
Version    : -
Charm      : -
Author     : D. Quehl (Qu); Siemens AG B Med CSA
Description : Initial release
}-----*/

/*{ Change header -----
Date       : 07. Jul 1996
Version    : 1.1
Charm      : 2041
Author     : D. Quehl (Qu); Siemens AG B Med CSA
Description : Changes in respect to code review component
            *osc - generic main - 03.jul.96 *
}-----*/

```

```

/*{ Compilation unit -----
Component      : CSA -
Name           : CsaGenericMainBas.cpp
Author        : (D. Quehl, BNEP11, +49 9131 84 2430); Siemens AG B MEd CSA
Language      : C/C++
Creation Date  : 19-JUN-1996
Test State    :
Description    : Implementation of the Generic Main functionality.
Requirement Key : os_process_mgr, os_os_test_stability
*/

```

Copyright (C) Siemens AG 1995 All Rights Reserved

```

/*} END */
-----*/

```

```

#pragma comment ( exestr, "e(f) CsaGenericMainBas.cpp 1.3 (10 Jul 1996) : Implementation Generic Main Base class" )

```

```

#include <os/CsaGenericMainBas.h>

```

```

/*{ Method -----

```

```

Name           : CsaGenericMainBas::CsaGenericMainBas
Description    : Default constructor for CsaGenericMainBas.
                  Prior to starting the dispatch loop the
                  command line arguments must be stored using
                  the setArgs() method.
                  The constructor calls the open() method.
Return         : none

```

```

/*} END Method */
-----*/
CsaGenericMainBas::CsaGenericMainBas(
    void          // takes no arguments
)

```

```

/*} END Method */
: argc_(0), argv_(0), initialized_(false)
{
    CSA_TRACE_IN ((CSA_OSC, "int CsaGenericMainBas::CsaGenericMainBas"));
    this->open();
}

```

```

/*{ Method -----

```

```

Name           : CsaGenericMainBas::CsaGenericMainBas
Description    : Constructor for CsaGenericMainBas.
                  Stores the command line arguments and calls the open()
                  method.
Return         : none

```

```

-----*/
CsaGenericMainBas::CsaGenericMainBas(
    int argc,      // argument count
    char *argv[]  // argument vector
)

```

```

/*} END Method */
: initialized_(false)
{
    CSA_TRACE_IN ((CSA_OSC, "int CsaGenericMainBas::CsaGenericMainBas"));
    this->setArgs(argc,argv);
    this->open();
}

```

```

/*{ Method -----

```

```

Name           : CsaGenericMainBas::open
Description    : Perform initialization steps for CsaGenericMainBas.
                  - Create our own thread manager
Return         : 0          success
                  -1         error

```

```

-----*/
int CsaGenericMainBas::open (
    void *p      // pointer for arbitrary use
)

```

```

/*} END Method */
{
    CSA_TRACE_IN ((CSA_OSC, "int CsaGenericMainBas::open"));
    if ((this->thrMgr_ = new ACE_Thread_Manager) == 0) {
        CSA_STATUS_GMAIN(NOWEM_NEW);
        return -1;
    }
    this->thr_mgr(thrMgr_);
    return 0;
}

```

```

/*{ Method -----

```

Name : CsaGenericMainBas::handle_signal
 Description : The handle_signal() method is called on an incoming SIGINT.
 handle_signal() calls endloop() to terminate dispatching.

Return : 0 success
 -1 error

```

-----*/
int CsaGenericMainBas::handle_signal (
    int signalnum // signal number that caused invocation
)
{
    /*] END Method */

    CSA_TRACE_IN ((CSA_OSC, "int CsaGenericMainBas::handle_signal"));

    if (this->thrMgr_ != 0)
        return this->endloop();

    CSA_STATUS_OBAIN(NOWM_NEW);
    return -1;
}

```

/*] Method

Name : CsaGenericMainBas::svc

Description : The svc() method implementation of the main dispatch loop.
 After starting the ACE_Service_Config daemon with the stored command line arguments the first hook method is called and - if this hook returns success - the dispatch loop invoked.
 Immediately after returning from the dispatch loop the second hook method is called to perform subclass specific cleanup actions while the ACE_Service_Config is still up.

Return : 0 success
 -1 error

```

-----*/
int CsaGenericMainBas::svc (
    void
)
{
    /*] END Method */

    CSA_TRACE_IN ((CSA_OSC, "int CsaGenericMainBas::svc"));

    // return error if thread manager creation failed in open()
    if (this->thrMgr_ == 0) {
        CSA_STATUS_OBAIN(NOWM_NEW);
    }
}

```

```

    return -1;
}

// Look the access to the dispatch loop
ACE_TSS_Guard<ACE_Thread_Mutex> mon (this->lock_);

// make us owner of the reactor
ACE_Service_Config::reactor()->owner(ACE_Thread::self());

// invoke the startHook
if (this->openHook() == -1)
    return -1;

// Start the daemon
ACE_SEH_TRY {
    if (this->daemon_.open (this->argc_, this->argv_) == -1) {
        CSA_STATUS_OBAIN(NOWM_REACTOR);
        return -1;
    }
}
ACE_SEH_EXCEPT(EXCEPT_EXECUTE_HANDLER) {
    cout << "Caught structured exception while opening Service_Config daemon" << endl;
}

// Register this to the reactor to receive SIGINT
if (ACE_Service_Config::reactor()->register_handler (SIGINT, this) == -1) {
    CSA_STATUS_OBAIN(REGISTER_HANDLER);
    (void)this->daemon_.close();
    return -1;
}

// Call the first hook; fall through if hook returns an error
if (this->loopHook1() >= 0) {
    initialized_ = true;
    ACE_SEH_TRY {
        ACE_Service_Config::run_event_loop ();
    }
    ACE_SEH_EXCEPT(EXCEPT_EXECUTE_HANDLER) {
        cout << "Caught structured exception while running Service_Config event loop" << endl;
    }
    (void)this->loopHook2();
}

int status;

// Close down the services of daemon
ACE_SEH_TRY {
    status = this->daemon_.close_svc();
}
ACE_SEH_EXCEPT(EXCEPT_EXECUTE_HANDLER) {
    cout << "Caught structured exception while closing down Service_Config daemon" << endl;
}

// invoke the close hook
(void)this->closeHook();
}

```

```

// Close down the memory of daemon
status = this->daemon._close_main();

initialized_ = false;
return status;
}

/*| Method -----*/
Name      : CsaGenericMainBas::dispatchLoop
Description : Invoke dispatch loop either synchronously (in the context
              of the caller thread) or asynchronously (using
              ACE_Task::activate() to create a separate thread).
Return     : 0      success
            -1     error
-----*/

int CsaGenericMainBas::dispatchLoop(
    bool async // flag for asynchronous invocation
)
{
    /*| END Method */

    // return error if thread manager creation failed in open()
    if (this->thrMgr == 0) {
        CSA_STATUS_CHAIN(NOWARN_NEW);
        return -1;
    }

    // check whether initialization was already set
    if (!initialized_) {
        // for synchronous operation just invoke svc method in the context
        // of the caller's thread.
        if (!async)
            return this->svc();

        // for asynchronous operation invoke svc method in the context
        // of a thread created by activate().
        else
            return this->activate();
    }

    // Return error if invoked twice.
    return -1;
}

/*| Method -----*/
Name      : CsaGenericMainBas::endLoop

```

```

Description : Stop dispatching.
Return     : 0      success
            -1     error
-----*/

int CsaGenericMainBas::endLoop(
    void
)
{
    /*| END Method */

    ACE_Service_Config::end_event_loop();
    return 0;
}

/*| Method -----*/
Name      : CsaGenericMainBas::sync
Description : Synchronize on dispatcher thread finished
Return     : none
-----*/

void CsaGenericMainBas::sync(
    void
)
{
    /*| END Method */

    if (this->thrMgr != 0)
        this->thrMgr->wait();
    ACE_Thread::exit(0);
}

/*| Change header -----*/
Date      : 26. Jun 1996
Version   : -
Charm     : -
Author    : D. Quehl (Qu); Siemens AG B Mad CSA
Description : Initial release
-----*/

/*| Change header -----*/
Date      : 07. Jul 1996

```

Version : 1.1
Charm : 2041
Author : D. Quehl (Qu); Siemens AG B Med CSA

Description : Changes in respect to code review component
"osc - generic main - 03.jul.96"

]-----*/

```

/*[ Compilation unit -----
Component      : CSA -
Name          : CsaGenericMain.h
Author        : (D. Quehl, BNEP11, +49 9131 84 2430); Siemens AG B Med CSA
Language      : C/C++
Creation Date  : 19-JUN-1996
Test State    :
Description    : Add on to Generic Main functionality.
                Definition of library startup/shutdown functionality.
Requirement Key : os_os_process_mgr

```

Copyright (C) Siemens AG 1995 All Rights Reserved

```

/*] END */
#ifndef CSAGENERICMAIN_H
#define CSAGENERICMAIN_H

#pragma comment ( exestr, "e(1) CsaGenericMain.h 1.4 (10 Jul 1996) test: Definition Generic Main" )

#include <os/CsaGenericMainBas.h>

// Header files of Singletons
#include <os/CsaSesam.h>
#include <os/CsaMscComm.h>
#include <os/CsaOsSystem.h>
#include <os/CsaMgtCmdIf.h>

/*[ Class -----
Name          : CsaGenericMain
Description    : CsaGenericMain is a friend class of a couple of
                basic libraries all designed using the singleton
                pattern with a private constructor/destructor and
                destroy() method. These singletons will be instantiated
                in the open method of CsaGenericMain and destroyed
                (in reverse order) in the close() method.

class CsaGenericMain : public CsaGenericMainBas {
public:
    // Constructor
    CsaGenericMain(int argc, char *argv[])

```

```

    : CsaGenericMainBas(argc, argv) {}

// The openhook() method is called by the svc() method and instantiates
// all singletons the generic main provides for the component DLLs.
virtual int openhook (void);

// The closehook() method is called by the svc() method to close up
// all singletons the generic main provided for the components
// (reverse order).
virtual int closehook (void);

// Instances of of singletons
CsaMscComm *commu_;
CsaSesam *sesam_;
CsaOsSystem *os_;
CsaMgtCmdIf *rti_;
};

/*] END Class */

#endif //ifndef CSAGENERICMAIN_H

/*[ Change header -----
Date          : 26. Jun 1996
Version       : -
Charm        : -
Author        : D. Quehl (Qui); Siemens AG B Med CSA
Description   : Initial release
]-----*/

```

```

/*{ Compilation unit -----
Component      : CSA -
Name           : CsaGenericMain.cpp
Author        : (D. Quehl, ENEP11, +49 9131 84 2430); Siemens AG B Med CSA
Language      : C/C++
Creation Date  : 19-JUN-1996
Test State    :
Description    : Add on to Generic Main functionality.
                Implementation of libray startup/run/ndown functionality.
Requirement Key : os_os_process_mgr, os_os_test_stability

```

Copyright (C) Siemens AG 1995 All Rights Reserved

```

/*} END */
-----*/

```

```

#pragma comment ( exstr, "e{#} CsaGenericMain.cpp 1.4 (10 Jul 1996; test: Implementation Generic Main" )

```

```

#include <osc/CsaGenericMain.h>

```

```

/*{ Method -----

```

```

Name           : CsaGenericMain::openHook
Description    : The openHook() method is called by the svc() method and
                instantiates all singletons the generic main provides
                for the component DLLs.

```

```

Return        : 0          success
               -1         error

```

```

int CsaGenericMain::openHook (
    void
)
    // takes no arguments

```

```

/*} END Method */

```

```

CSA_TRACE_IN (CSA_OSC, "int CsaGenericMain::openHook*");

```

```

this->sesam_ = CsaSesam::sesam();
if (this->sesam_ == 0) {
    CSA_STATUS_GMAIN(NOMEM_NEW);
}
this->commu_ = CsaComm::commu();
if (this->commu_ == 0) {

```

```

    CSA_STATUS_GMAIN(NOMEM_NEW);
}
this->os_ = CsaOsSystem::csaOsSystem();
if (this->os_ == 0) {
    CSA_STATUS_GMAIN(NOMEM_NEW);
}
this->rti_ = CsaPmgtCmdIf::csaPmgtCmdIf();
if (this->rti_ == 0) {
    CSA_STATUS_GMAIN(NOMEM_NEW);
}
return 0;
}

```

```

/*{ Method -----

```

```

Name           : CsaGenericMain::closeHook

```

```

Description    : The closeHook() method is called by the svc() method and
                destroys all singletons the generic main provided for the
                components (reverse order).

```

```

Return        : 0          success
               -1         error

```

```

int CsaGenericMain::closeHook (
    void
)
    // takes no arguments

```

```

/*} END Method */

```

```

    // close down singletons

```

```

    // cout << "CsaGenericMain::closeHook>>> closing down libraries" << endl;

```

```

    // cout << "CsaPmgtCmdIf" << endl;
    this->rti_>destroy();

```

```

    // cout << "CsaOsSystem" << endl;
    this->os_>destroy();

```

```

    // cout << "CsaMsc" << endl;
    this->commu_>destroy();

```

```

    // cout << "CsaSesam" << endl;
    this->sesam_>destroy();

```

```

    // cout << "CsaGenericMain::closeHook>>> closing down libraries finished" << endl;

```

```

    return 0;
}

```

```

/*{ Change header -----

```

Date : 26. Jun 1996
Version : -
Cham : -
Author : D. Quehl (Qu); Siemens AG B Med CSA
Description : Initial release

]------*/

/*[Change header -----

Date : 07. Jul 1996
Version : 1.1
Cham : 2041
Author : D. Quehl (Qu); Siemens AG B Med CSA

Description : Changes in respect to code review component
"osc - generic main - 03.jul.96"

]------*/

5. Welche Dienststellen sind an der Erfindung interessiert: _____

6. Wurde die Erfindung bereits erprobt (Durchführung von Versuchen, Anfertigung von Mustern)?

☐ nein ☐ ja, Ergebnis: _____

7. Für welche Erzeugnisse ist die Erfindung anwendbar? _____

8. Ist die Anwendung der Erfindung vorgesehen?

☐ nein ☐ ja, bei: _____

9. Ist ein auf der Erfindung beruhendes Erzeugnis geliefert oder ist eine Lieferung beabsichtigt?

☐ nein ☐ ja, am _____ Bezeichnung des Erzeugnisses: _____

10. Ist eine Veröffentlichung der Erfindung beabsichtigt oder bereits erfolgt?

☐ nein ☐ ja, am _____ in Buch, Zeitschrift: _____

11. Ist eine Mitteilung der Erfindung an Firmenfremde beabsichtigt oder bereits erfolgt?

☐ nein ☐ ja, am _____ an _____

12. Angaben zur Person des/der Erfinder/s (für jeden Erfinder ist eine senkrechte Spalte vorgesehen):

Name	Dr. Schmidt			
Vorname	Douglas C.			
akad. Grad/Titel/Beruf				
Dienstanschrift	Washington Univ. Depart. m. of Comp. Science			
Tätigkeit/Stellung im Betrieb (z.B. Lehrvorseher u.ä.)	Assistant Professor			
Hausanruf	/			
Staatsangehörigkeit	US			
Postleitzahl, Wohnort	Campus Box 1045 One Brookings Drive St. Louis, Missouri			
Straße, Haus-Nr.				
Geburtsdatum	18.7.1962			
Abrechnende Personaldienststelle oder APD-Nr. *)				
Personalnummer *)	/			
13. Liegt die Erfindung auf a) Ihrem Arbeitsgebiet? b) einem anderen Arbeitsgebiet Ihres Arbeitgebers?	<input type="checkbox"/> ja <input type="checkbox"/> nein <input type="checkbox"/> ja <input type="checkbox"/> nein	<input type="checkbox"/> ja <input type="checkbox"/> nein <input type="checkbox"/> ja <input type="checkbox"/> nein	<input type="checkbox"/> ja <input type="checkbox"/> nein <input type="checkbox"/> ja <input type="checkbox"/> nein	<input type="checkbox"/> ja <input type="checkbox"/> nein <input type="checkbox"/> ja <input type="checkbox"/> nein
14. Welchen Anteil an der Erfindung haben Sie?	20%			
15. Wurde die Erfindung als V V gemeldet?	<input type="checkbox"/> ja <input checked="" type="checkbox"/> nein	<input type="checkbox"/> ja <input type="checkbox"/> nein	<input type="checkbox"/> ja <input type="checkbox"/> nein	<input type="checkbox"/> ja <input type="checkbox"/> nein
16. Falls Sie die Erfindung als freie Erfindung ansehen, bitte begründen:				
17. Meines/unsere Wissens sind keine weiteren Personen an der Erfindung beteiligt.	Douglas C. Schmidt Unterschrift	Unterschrift	Unterschrift	Unterschrift
Vermerke der GR PA				

*) Bitte aus Firmenausweis oder Gehaltsabrechnung entnehmen

5. Welche Dienststellen sind an der Erfindung interessiert? _____

Aktenzeichen der GR PA

6. Wurde die Erfindung bereits erprobt (Durchführung von Versuchen, Anfertigung von Mustern)?

86E 8583

☐ nein ☒ ja, Ergebnis: gut

7. Für welche Erzeugnisse ist die Erfindung anwendbar? für jegliche Art von Main(-) - Programmen

8. Ist die Anwendung der Erfindung vorgesehen?

☐ nein ☒ ja, bei: MED CSA Projekt

9. Ist ein auf der Erfindung beruhendes Erzeugnis geliefert oder ist eine Lieferung beabsichtigt?

☐ nein ☒ ja, am 01/97 Bezeichnung des Erzeugnisses: _____

10. Ist eine Veröffentlichung der Erfindung beabsichtigt oder bereits erfolgt?

☒ nein ☐ ja, am _____ in Buch, Zeitschrift: _____

11. Ist eine Mitteilung der Erfindung an Firmenfremde beabsichtigt oder bereits erfolgt?

☐ nein ☐ ja, am _____ an _____

12. Angaben zur Person des/der Erfinder/s (für jeden Erfinder ist eine senkrechte Spalte vorgesehen):

Name	Dorn	Quell	Becher	Schäuf
Vorname	Karlheinz	Dieter	Detlef	Christian
akad. Grad/Titel/Beruf	Dipl.-Inform.	Dipl.-Ing.	Dipl.-Ing.	Dipl. Ing
Dienstanschrift	MED BNSW-ART	MED BNSW-ART	MED BNSW-ART	MED BNSW-ART
Tätigkeit/Stellung im Betrieb (z.B. Laborvorsteher u.ä.)	Systementw.	SW-Entwickler	SW-Entwickler	SW-Entwickler
Hausanruf	8187	2430	8305	7218
Staatsangehörigkeit	deutsch	deutsch	deutsch	deutsch
Postleitzahl, Wohnort	90562 Velchreuth	91052 Glangen	91096 Möhrndorf	91074 Herzogenaurach
Straße, Haus-Nr.	Erlenstr. 29	Nürnbergstr. 83	Wasserwerkstr. 10	Siedlung 10
Geburtsdatum	18.9.56	12.4.55	19.1.60	27.06.43
Abrechnende Personaldienststelle oder APD-Nr. *)	052	052	052	052
Personalnummer *)	465559	174637	420611	194129
13. Liegt die Erfindung auf a) Ihrem Arbeitsgebiet? b) einem anderen Arbeitsgebiet Ihres Arbeitgebers?	<input type="checkbox"/> ja <input type="checkbox"/> nein <input type="checkbox"/> ja <input type="checkbox"/> nein	<input type="checkbox"/> ja <input type="checkbox"/> nein <input type="checkbox"/> ja <input type="checkbox"/> nein	<input type="checkbox"/> ja <input type="checkbox"/> nein <input type="checkbox"/> ja <input type="checkbox"/> nein	<input type="checkbox"/> ja <input type="checkbox"/> nein <input type="checkbox"/> ja <input type="checkbox"/> nein
14. Welchen Anteil an der Erfindung haben Sie?	20%	20%	20%	20%
15. Wurde die Erfindung als V V gemeldet?	<input type="checkbox"/> ja <input checked="" type="checkbox"/> nein	<input type="checkbox"/> ja <input checked="" type="checkbox"/> nein	<input type="checkbox"/> ja <input checked="" type="checkbox"/> nein	<input type="checkbox"/> ja <input checked="" type="checkbox"/> nein
16. Falls Sie die Erfindung als freie Erfindung ansehen, bitte begründen:				
17. Meines/unseres Wissens sind keine weiteren Personen an der Erfindung beteiligt.	<u>[Signature]</u> Unterschrift	<u>[Signature]</u> Unterschrift	<u>[Signature]</u> Unterschrift	<u>[Signature]</u> Unterschrift
Vermerke der GR PA				

*) Bitte aus Firmenausweis oder Gehaltsabrechnung entnehmen